C-String

- Inherited from C language
- An array of characters ends with a null character
  \[\text{char[6]} = \text{“hello”; } //\text{c-string}\]

  | h | e | l | l | o | \0 |

A c-string is not just an array of characters

\[\text{char[5]} = \text{“hello”; } //\text{c-characters}\]

  | h | e | l | l | o |
String

- String is a class build into the C++ library.

- String has predefined functions contained within the class which we can use for our convenience to do string manipulations.

String Functions

- Two types of functions
  - Member functions
    Call on behave of a string data
  - Non-member functions
    Take a string as it’s argument
Non-member overloads

- operator>>
- operator<<
- operator+
- getline

#include <iostream>
using namespace std;

int main()
{
    string str1, str2;
    cin >> str1;
    cout << str1;
    cout << str1; //concatenate 2 strings
    getline(cin, str1);
    return 0;
}
Member Functions

operator[]
length()
insert()
substr()
find()
rfind()
c_str()
compare()
We may see string as an array of characters.
String str1 = “hello”;

```
0 1 2 3 4
h e l l o
```
cout << str1[0]; // first character of str1
Member Function-length()

To identify the length of the string.
String st = “hello”;

```
<table>
<thead>
<tr>
<th>h</th>
<th>e</th>
<th>l</th>
<th>l</th>
<th>o</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
```

cout << “Characters count: “ << st.length();
**Member Function-`insert()`**

- String library also allows us to insert some text into part of the string, instead of append the text to the end of a string. We use the `insert` function.

**Model:**
```
std::string str_var.insert( index_pos, text_tobe_insert );
```

- `index_pos`: the starting position in the `str_var` where you want the text to go, then push all the characters after the `index_pos` in the original `str_var` back after the inserted text
- `text_tobe_insert`: the text you want to insert into the `str_var` string

```
string st = "goodbye";
```

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>g</td>
<td>o</td>
<td>o</td>
<td>d</td>
<td>b</td>
<td>y</td>
<td>e</td>
</tr>
</tbody>
</table>

```
st.insert(4, " ");
```

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>g</td>
<td>o</td>
<td>o</td>
<td>d</td>
<td>b</td>
<td>y</td>
<td>e</td>
<td>Page 9</td>
</tr>
</tbody>
</table>
Member Function-\texttt{substr()}

- To get a partial string from the original string.
- This function has 2 signature

\texttt{Model 1: str\_var.substr( start\_index );}
\texttt{Model 2: str\_var.substr( start\_index, char\_count );}

String st = “good bye”;

\begin{tabular}{cccccccc}
  g & o & o & d &  & b & y & e \\
  0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\end{tabular}

\texttt{cout << st.substr(3); //d bye}
\texttt{cout << st.substr(5, 2) //by}
Member Function-find()

- Searches the string for the **first** occurrence of the sequence specified by its arguments.
- It’s **not** a boolean function.

Model 1: str_var.substr(partial_string_to_search);

String st = “good bye”;

<table>
<thead>
<tr>
<th></th>
<th>g</th>
<th>o</th>
<th>o</th>
<th>d</th>
<th>b</th>
<th>y</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>

cout << “find first o at index: ” << st.find(“o”);  // output 1
Member Function-rfind()

- Searches the string for the **last** occurrence of the sequence specified by its arguments.
- It’s **not** a boolean function.

Model 1: str_var.substr(partial_string_to_search);

String st = “good bye”;

```
g o o d b y e
```

```
0 1 2 3 4 5 6 7
```

cout << “find last o at index: ” << st.find(“o”);
//output 2